Robot Description

Robotics Club

University of Patras
Table of contents

1. Tools

2. Robots

3. Development
   - Installation of ROS Hydro
   - Catkin workspace
   - Rosbuild workspace
   - Catkin Packages

4. Assignment
Tools

- Ubuntu/Kubuntu/Xubuntu/Lubuntu/Ubuntu GNOME (LTS preferable, 12.04)
- ROS Hydro installation
- Gazebo installation
- Bitbucket Account
- Code Editor (QT Creator/Sublime Text/Spyder)
Tools

- Ubuntu/Kubuntu/Xubuntu/Lubuntu/Ubuntu GNOME (LTS preferable, 12.04)
- ROS Hydro installation
- Gazebo installation
- Bitbucket Account
- Code Editor (QT Creator/Sublime Text/Spyder)

Optional
- Dropbox account
- Github account
- Lots and lots of time...
Bitbucket
Create a bitbucket account using your academic e-mail (you get unlimited private repositories) Ask to be joined to the robotics-club repo as well as the p3-at repo

ROS Installation
Follow the instructions in the robotics-club repo wiki to install ROS Hydro
https://bitbucket.org/Progtologist/robotics-club/wiki/Hydro%20Useful%20Information
Robotics Club  \{ master, hydro-devel, groovy-devel, fuerte-devel \}

p3-at  \{ master, hydro-devel, groovy-devel \}

Robotics Club - ROS Team  \{ master \}
Getting to know

Online Tutorials

- Linux
- ROS
- Gazebo
- Git
Getting to know

Online Tutorials
- Linux
- ROS
- Gazebo
- Git

Robotics Club E-Books
- LINUX.pdf
- ROS.pdf
- GIT.pdf
- PhD Pandora Team.pdf
Table of contents

1 Tools

2 Robots

3 Development
   • Installation of ROS Hydro
   • Catkin workspace
   • Rosbuild workspace
   • Catkin Packages

4 Assignment
Our robots
How we work

Gazebo

Initially we work with simulation models of the Pioneer P3-AT with a 5DoF Manipulator and simulated sensors. We use the packages available in the p3-at repository, branch master.

How-to

In your catkin_ws folder

cd src

wstool set p3-at –git git@bitbucket.org:Progtologist/p3-at.git

wstool update p3-at

cd..

catkin_make
How we work

Develop in Robotics Club - ROS Team

In your catkin_ws folder
cd src
wstool set robotics_club-ros_team --git
git@bitbucket.org:Progtologist/robotics-club-ros-team.git -v
hydro-devel
wstool update robotics_club-ros_team
cd..
catkin_make

Creating Packages

You cannot (and should not) push in the p3-at repo
Create your package in the robotics_club-ros_team repo
Always use a testing branch before pushing to master
Never push to master if your code is not compiling!
Table of contents

1 Tools

2 Robots

3 Development
   • Installation of ROS Hydro
   • Catkin workspace
   • Rosbuild workspace
   • Catkin Packages

4 Assignment
Languages/Frameworks

- C++, Python, XML, YAML, CMake
- Messages/Services
- Drivers/Controllers
- Libraries (actionlib, pluginlib, TF, nodelet, RViz, navigation, PCL, RobotModel)
- GUI development using RQT
- Debugging tools (GDB, Oprofile, Valgrind, Test tools)
- Documentation (API, wiki)
- Releasing, Maintaining, Standardising

Links at Developers Guide and ROS API
First you will need to install ROS - Hydro

**ROS Installation**

```bash
sudo sh -c 'echo "deb http://packages.ros.org/ros/ubuntu precise main" > /etc/apt/sources.list.d/ros-latest.list'
wget http://packages.ros.org/ros.key -O - | sudo apt-key add -
sudo apt-get update && sudo apt-get install ros-hydro-desktop-full
sudo apt-get install ros-hydro-robot-state-publisher
ros-hydro-pr2-controller-manager ros-hydro-gazebo-ros-pkgs
ros-hydro-gazebo-ros-control ros-hydro-ros-control
sudo rosdep init && rosdep update
```

Then you need to install some general usage tools

**General Tools**

```bash
sudo apt-get install python-rosinstall python-wstool
python-bloom python-vcstools python-rospkg python-rosdep
```
The catkin build system has several attributes
The catkin build system has several attributes

- Packages are not built separately
The catkin build system has several attributes

- Packages are not built separately
- Distributing packages is very easy
The catkin build system has several attributes

- Packages are not built separately
- Distributing packages is very easy
- Automated build system for *.deb files for everyone
The catkin build system has several attributes

- Packages are not built separately
- Distributing packages is very easy
- Automated build system for *.deb files for everyone
- Developing and revising source code is intuitive
The catkin build system has several attributes

- Packages are not built separately
- Distributing packages is very easy
- Automated build system for *.deb files for everyone
- Developing and revising source code is intuitive
- General framework for all VCS
The catkin build system has several attributes

- Packages are not built separately
- Distributing packages is very easy
- Automated build system for *.deb files for everyone
- Developing and revising source code is intuitive
- General framework for all VCS
- Source separated from distribution
Workspace Architecture

Source your basic environment

source /opt/ros/hydro/setup.sh

Create a new Catkin Workspace

Building a new catkin workspace

mkdir -p ~/hydro/catkin_ws/src
cd ~/hydro/catkin_ws/src
catkin_init_workspace
wstool init
cd ..
catkin_make
A catkin workspace is a folder where you modify, build, and install catkin packages. The following is the recommended and typical catkin workspace layout:

A catkin workspace can contain up to four different spaces which each serve a different role in the software development process.
The source space contains the source code of catkin packages. This is where you can extract/checkout/clone source code for the packages you want to build. Each folder within the source space contains one or more catkin packages. This space should remain unchanged by configuring, building, or installing.
The root of the source space contains a symbolic link to catkin’s boiler-plate ’toplevel’ CMakeLists.txt file. This file is invoked by CMake during the configuration of the catkin projects in the workspace.
The build space is where CMake is invoked to build the catkin packages in the source space. CMake and catkin keep their cache information and other intermediate files here. The build space does not have to be contained within the workspace nor does it have to be outside of the source space, but this is recommended.
The development space (or devel space) is where built targets are placed prior to being installed. The way targets are organized in the devel space is the same as their layout when they are installed. This provides a useful testing and development environment which does not require invoking the installation step. The location of the devel space is controlled by a catkin specific CMake variable called CATKIN_DEVEL_PREFIX, and it defaults to `<build space>/develspace`. This is the default behavior because it might be confusing to CMake users if they invoked cmake .. in a build folder and that modified things outside of the current directory. It is recommended, however, to set the devel space directory to be a peer of the build space directory.
Once targets are built, they can be installed into the install space by invoking the install target, usually with make install. The install space does not have to be contained within the workspace.

Since the install space is set by the CMAKE_INSTALL_PREFIX, it defaults to /usr/local, which you should not use (because uninstall is near-impossible, and using multiple ROS distributions does not work neither). One of the main reasons for moving away from rosbuild is that it lacked a proper install target. This prevented ros packages from being used like canonical system packages and made distribution more difficult. Catkin encourages standard compliant install layouts via CMake. Therefore, when you are ready, you can install your packages to the system for others to use, or you can more easily package your packages for distribution systems like deb or rpm.
A rosbuilt workspace is the folder where you modify, build, and install rosbuilt packages. It is the typical build system used by ros up to fuerte. The source, build and devel folders are not separated from the packages.

Create a rosbuilt workspace

cd ~/hydro
mkdir rosbuilt_ws
cd rosbuilt_ws
rosws init ../catkin_ws/devel
wstool init

```
catkin_ws/       --CATKIN WORKSPACE
    ...
rosbuild_ws      --ROSBUILD WORKSPACE
    setup.sh
    setup.bash
    setup.zsh
    .rosinstall
```
Overlaying with Catkin

Instead of creating a rosbuilt workspace that refers to your installed ROS distribution (e.g. /opt/ros/hydro), we refered to this catkin workspace. You can source the generated rosbuilt_ws/setup.sh files to work with both the rosbuilt and the catkin workspace. Else nothing changes. If you have a running rosbuilt workspace and want to change what it refers to, you will have to change your .rosinstall file – setup-file: {local-name: /home/<user>/catkin_ws/devel/setup.sh} Just change the path in this line of your .rosinstall file to point to the setup.sh file you want to use. Note it is not recommended to switch to another ROS distribution, but you could well change from /home/<user>/catkin_ws/devel/setup.sh to /home/<user>/catkin_ws/install/setup.sh
For a package to be considered a catkin package it must meet a few requirements:
– The package must contain a catkin compliant package.xml file
The package.xml file provides meta information about the package
– The package must contain a CMakeLists.txt which uses catkin
The exception to this rule is that metapackages do not have a CMakeLists.txt file There can be no more than one package in each folder This means no nested packages nor multiple packages sharing the same directory
Creating a new package

Package creation

```
cd ~/hydro/catkin_ws/src
catkin_create_pkg beginner_tutorials std_msgs rospy roscpp
```
Your CMakeLists.txt file MUST follow this format otherwise your packages will not build correctly.
- Required CMake Version (cmake_minimum_required)
- Package Name (project())
- Find other CMake/Catkin packages needed for build (find_package())
- Message/Service/Action Generators (add_message_files(), add_service_files(), add_action_files())
- Invoke message/service/action generation (generate_messages())
- Specify package build info export (catkin_package())
- Libraries/Executables to build (add_library()/add_executable()/target_link_libraries())
- Tests to build (catkin_add_gtest())
- Install rules (install())
cmake_minimum_required(VERSION 2.8.3)
project(phidgets)
find_package(catkin REQUIRED COMPONENTS roscpp std_msgs message_generation tf sensor_msgs)
set(CMAKE_EXE_LINKER_FLAGS "-lw1,lpidget21")
add_message_files(FILES servo_params.msg stepper_params.msg)
add_service_files(FILES servo_reference.srv servo_reference_joints.srv)
generate_messages(DEPENDENCIES std_msgs sensor_msgs)
catkin_package(
    DEPENDS robot_pose_ekf
    CATKIN_DEPENDS message_runtime sensor_msgs tf roscpp std_msgs
    INCLUDE_DIRS
    LIBRARIES
)
include_directories(include \${catkin_INCLUDE_DIRS})
add_executable(advanced_servo src/advanced_servo.cpp)
target_link_libraries(advanced_servo \${catkin_LIBRARIES})
target_link_libraries(advanced_servo phidget21)
add_dependencies(advanced_servo phidgets_gencpp)
add_executable(stepper src/stepper.cpp)
target_link_libraries(stepper \${catkin_LIBRARIES})
target_link_libraries(stepper phidget21)
add_dependencies(stepper phidgets_gencpp)
add_executable(stepper_client src/stepper_client.cpp)
target_link_libraries(stepper_client \${catkin_LIBRARIES})
target_link_libraries(stepper_client phidget21)
add_dependencies(stepper_client phidgets_gencpp)
add_executable(Servo_Joint_Test src/Servo_Joint_Test.cpp)
target_link_libraries(Servo_Joint_Test \${catkin_LIBRARIES})
add_dependencies(Servo_Joint_Test phidgets_gencpp)
add_executable(spatial_IMU src/spatial_IMU.cpp)
target_link_libraries(spatial_IMU \${catkin_LIBRARIES})
target_link_libraries(spatial_IMU phidget21)
Table of contents

1 Tools

2 Robots

3 Development
   • Installation of ROS Hydro
   • Catkin workspace
   • Rosbuild workspace
   • Catkin Packages

4 Assignment
Initial Goals

1. Install your favorite ubuntu distro
Initial Goals

1. Install your favorite ubuntu distro
2. Install ROS Hydro
Initial Goals

1. Install your favorite Ubuntu distro
2. Install ROS Hydro
3. Create your workspace
Initial Goals

1. Install your favorite ubuntu distro
2. Install ROS Hydro
3. Create your workspace
4. Add the repos
Initial Goals

1. Install your favorite ubuntu distro
2. Install ROS Hydro
3. Create your workspace
4. Add the repos
5. Create a catkin package for ROS (in ROS Team)
Initial Goals

1. Install your favorite ubuntu distro
2. Install ROS Hydro
3. Create your workspace
4. Add the repos
5. Create a catkin package for ROS (in ROS Team)
6. Push the package in the repo
Keyboard-Teleop

Create a package named ‘keyteleop_p3at’
The package created will teleoperate the p3at simulated robot with the PArm using the keyboard
All the joints must be able to move.
Presentation Goals

Presentations

Prepare for next meeting!

- Present TF/TF2
- Present URDF
- Present ActionLib